Search Algorithms for Improved Transportation Security under Covid-19 Lockdown

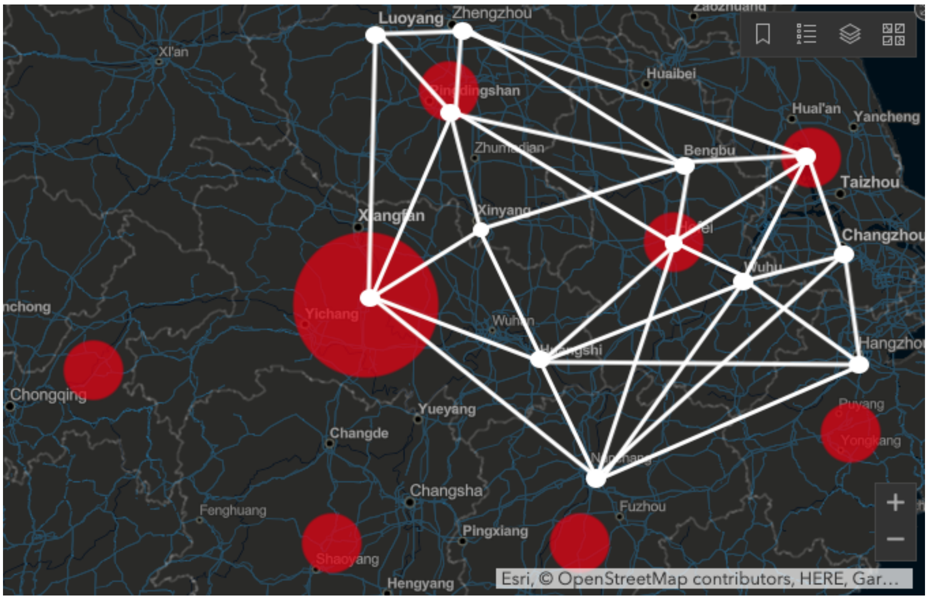
Diego Calanzone
University of Leeds, UK

02-27-2020

---

1 Real World Scenario

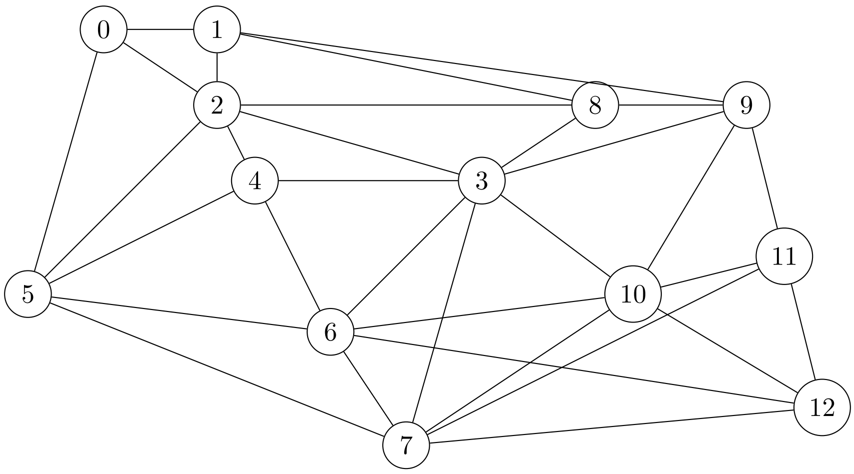
We consider as datasource *The 2019 Novel Coronavirus COVID-19 (2019-nCoV) Data Repository provided by Johns Hopkins CSSE*. In this demonstration we assume to compute a safe path from **Luoyang** to **Changzhou**. The subject will drive through risk-safe cities on the shortest path.



1. Eastern China Map with the problem graph. Red spots are Confirmed Cases.

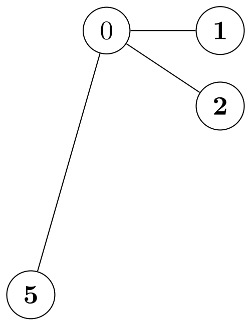
2 State representation

The possible city paths are represented by the given graph at the beginning:



3 Possible Actions

Each possible action is represented by the index of a *neighbour node*.



$$neighbour_0 = [\{1\}, \{2\}, \{5\}]$$

```python
def corona_possible_actions(state):

    possible_nodes = []

    # Pick all the nodes in the nearby list
    for node_index in graph[state][2]:
        possible_nodes.append(graph[node_index])

    return possible_nodes
```

4 Heuristic function

The considered heuristic function is composed by the weighted average of the node-to-node distance and the infection rate:

$$\textit{Infection Rate} = \frac{\textit{Infection Cases}}{\textit{City Population}} * \textit{GRANULARITY}$$

$$\textit{Node-to-node distance} = \sqrt{(Node_{1x} - Node_{2x})^2 + (\mathrm{Node}_{1y} - Node_{2y})^2}$$

$$\textit{h(state)} = \frac{(\textit{Node-to-node distance} * w_d) + (\textit{Infection Rate} * w_i)}{w_d + w_i}$$

```python
def node_infection_rate(node):

    # % of infected overall
    return (((Decimal(graph[node][0][1])) / Decimal(graph[node][0][0])) * GRANULARITY
        )

def node_to_node_distance(node1, node2):

    distance = sqrt(
        pow((graph[node1][1][0] - graph[node2][1][0]), 2) +
        pow((graph[node1][1][1] - graph[node2][1][1]), 2)
    )

    return distance

def corona_h_cost(state):

    # Distance cost
    distance = Decimal(node_to_node_distance(state, goal_node))

    # Infection cost
    ir = node_infection_rate(state)

    avg_weighted = ((distance * DIST_WEIGHT) + (ir * IR_WEIGHT)) / (DIST_WEIGHT +
        IR_WEIGHT)

    return avg_weighted
```

5 Goal State

The goal is reached when the destination node is reached. From the real problem instance, the algorithm computes the safest and cheapest path:

$$\text{Path}_{Luoyang-Changzhou} = [['Zhengzhou'], ['Bengbu'], ['Hefei'], ['Wuhu'], ['Changzhou']]$$

$$\text{Path}_{0-11} = [[1], [8], [3], [10], [11]]$$